King Saud University
College of computer and Information Sciences
CSC 311 – Design and Analysis of Algorithms

جامعة الملك سعود
كلية علوم الحاسب والمعلومات
311 عال –

Midterm Exam, Fall 2022 | Wednesday Oct 5th, 2022 | Exam time: 08:00-9:30 P.M.

Student's name: [redacted] Section: 9:15

43/50

Problem 1 (11 p[redacted]

(a) Give the following functions a number in order of increasing asymptotic growth rate. If two functions have the same asymptotic growth rate, give them the same number. [3 pts]

	Function	Rank
$\log n$	$3\ \log n + \log\log\log\ (n^3)$	1
$\log n$	$5 \lg n^5$	1
$n^3 >$	$\frac{n^3}{\lg n}$	2
n^3	$n^3 + 6\ log\ n - 5$	3
n^4	n^4 $16^{\log n} + 3\ n^2$	4
4^n	4^{3n}	5

(b) Using the definition of θ, find $g(n)$, C_1, C_2, and n_0 in the following: [4 pts]

$$6n^4 - 3n^3 + n\ log\ n \in \theta(g(n))$$

$O(n^4) = 6n^4 - 3n^3 + n\log n \leq 10n^4$ $\quad n_0 = 1$

$C_1 = 10, \quad n_{01} = 1$

$\Omega(n^4) = 6n^4 - 3n^3 + n\log n \geq 5n^4$ $\quad n_{0_2} = \frac{4}{6-5} = 4$

$C_2 = 5, \quad n_{02} = 4$

$\Theta(n^4)$
- $C_1 = 10$
- $C_2 = 5$
- n_0 $\max(n_{01}, n_{02}) = 4$

[4 pts]

(c) Prove that $log(n!)$ is $O(n\ log\ n)$.

$n! = n-1 \times n-2 \times n-3 ----$ complixity $n!$

So ~~..~~ log is minimize the complixity to $O(n \log n)$

how? -3

Problem 2 (7 points) 7/7

Consider the pseudo-code below:

```
S(A[0..n-1])
{
  L =0;        1
  R = n-1;     1
  While (L < R) n+1
  {
    if (A[L] mod 2 == 0) n
      L++;  n
    else if (A[R] mod 2 == 1) ——
      R--; ——
    Else ——
      Swap(A[L], A[R]); ——
  }
}
```

for(i=0 ; i<R ; L++) $\sum_{i=0}^{n-1} 1 + 1 = n-1+1+1 = n+1$

$T(n) = 3n+3$

a- Which problem does this algorithm solve?

It's sort the array by arrange All the even numbers in the first ~~then~~ before any odd number(~~...~~)

b- What is the time complexity (*Theta (?)*) of the following algorithm? Prove your answer (explain each step).

$T(n) = 3n+3$ $\qquad \Theta(n)$

$O(n) \rightarrow \begin{cases} c = 6 \\ n_0 = 1 \end{cases}$

$\Omega(n) = \begin{cases} c = 2 \\ n_0 = \frac{3}{3-2} = 3 \end{cases}$

$\Theta(n) \rightarrow \begin{cases} c_1 = 6 \\ c_2 = 2 \\ n_0 = 3 \end{cases}$

King Saud University
College of computer and Information Sciences
CSC 311 – Design and Analysis of Algorithms

جامعة الملك سعود
كلية علوم الحاسب والمعلومات
311 عال –

Problem 3 (9 points)

9/9

Consider the pseudo-code below, and assume the array A[..] is sorted in an increasing order:

```
S(A[..], key, imin, imax)
{
  if (imax < imin)      1
       return KEY_NOT_FOUND; 1
  else
    {
     imid = (imin+imax)/2;     1

     if (A[imid] > key)        1
       return S(A, key, imin, imid-1); T(n/2)
     else if (A[imid] < key)    —
       return S(A, key, imid+1, imax); —
     else   —
       return imid;  —
    }
}
```

$T(n) = T(\frac{n}{2}) + 4$

a- Which problem does this algorithm solve? ~~the element and return the mid~~.

~~[illegible]~~ Bainry search to find the mid element

b- What is the design technique used in this solution. Explain your answer.

Devid and concer technique

because we need to devid the element into two array each time and then return the result

$n^{\log_2 1}$ 4

c- What is the time complexity of the following algorithm? Prove your answer.

$T(n) = T(\frac{n}{2}) + 4$

$T(\frac{n}{2}) = [T(\frac{n}{4}) + 4] + 4$

$T(\frac{n}{4}) = T(\frac{n}{16}) + 4 + 4 + 4$

after k step : $T(\frac{n}{2^k}) + 4k$

$\frac{n}{2^k} = 1$

$n = 2^k$

$k = \log n$

$T(1) + 4\log n \Rightarrow 1 + 4\log n$

$O(\log n)$

$c = 5 \quad n_0 = 1$

Problem 4 (7 points) 5/7

Consider the following recurrence relation:

$$T(n) = 4T(\frac{n}{4}) + 3n.$$

Solve this recurrence relation using recursive substitutions. Find g(n), where **T(n) = O(g(n)).**

next page

$T(n) = \underbrace{4\,T(\frac{n}{4})} + 3n$

$T(\frac{n}{4}) = 4[4\,T(\frac{n}{16}) + 3\frac{n}{4}] + 3n = \underbrace{16\,T(\frac{n}{16}) + 12\frac{n}{4} + 3n}$

$T(\frac{n}{16}) = 16[4\,T(\frac{n}{64}) + 3\frac{n}{16}] + 12\frac{n}{4} + 3n$

$= \underbrace{64\,T(\frac{n}{64}) + 64\frac{n}{16} + 12\frac{n}{4} + 3n}$ ✓

after k: $4^k\,T(\frac{n}{4^k}) + 3n\sum_{i=0}^{k-1}\frac{4^i}{4^i}$ −2

Stop

$\frac{n}{4^k} = 1$ $\quad = 4^{\log n}\,T(1) + 3n$

$n = 4^k$

$k = \log n$ $\quad = n^2 + 3n$

6/6 $O(n^2)$ $\quad c = 4$

$n_0 = 1$

Problem 5 (6 points)

Solve the following recurrence using the Master theorem by giving tight θ-notation bounds. Justify your answers.

(a) T(n) =3T(n/3) + 3 log^3 (n) $\quad a = 3 \quad b = 3 \quad f(n) = 3\log^3(n)$

$3\log^3(n) \ll n^{\log_3 3} = n$

~~[illegible]~~ $\Theta(n)$ ✓

~~[illegible]~~

Problem 6 (10 points) 8/10

There are *n* parking spots numbered from 1 to *n* and you are told that there are *k* cars in the first *k* spots (one car in each spot), and all other spots are empty. How to find the value of *k*?

(a) Suggest **TWO** algorithm design techniques and give a high-level description of the **TWO** algorithms to solve the problem (find *k*)?

Prote force techniques

```
findk( A[1...n])
~~t≤o~~
 for(i←1...n)        n+1
     if(A[i] empty)  n
       retun i;      1
     i++;            n
```

$T(n) = 3n + 2$

Deide and concere

```
findk(A[1..n])
 mid ← n/2            1
     If(mid empty)    1
       findk(A[1.. mid-1]   T(n/2)
     return mid       1
```

else??

-2

$T(n) = T(\frac{n}{2}) + 3$

~~$T(n) = 1 + 3\log n$~~

(b) Analyze the complexity of your **TWO** algorithms?

① $T(n) = 3n+2$

$O(n)$ $c=5$, $n_0=1$ ✓

② $T(n) = T(\frac{n}{2})+3$

$T(n) \leq 1+3\log n$

$O(\log n)$

$c = 4$

$n_0 = 1$